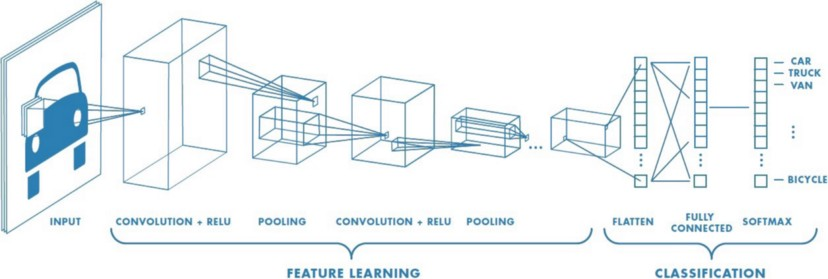
**AIDS Lab**

**EXPERIMENT NO. 9**

**Aim**: Implementing Deep Learning Application using CNN.

**Theory**:

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

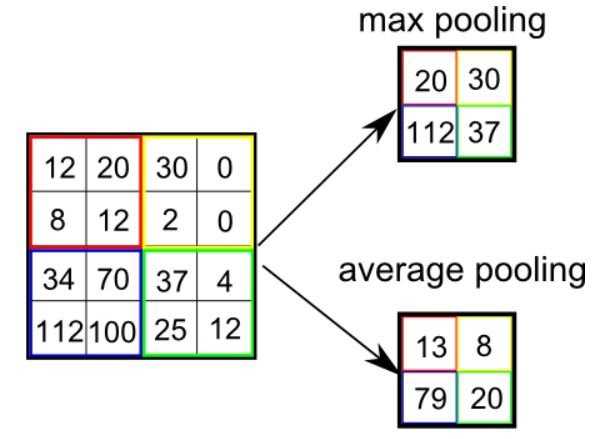


Convolutional Neural Network Design:

1. The construction of a convolutional neural network is a multi-layered feed-forward neural network, made by assembling many unseen layers on top of each other in a particular order.
2. It is the sequential design that gives permission to CNN to learn hierarchical attributes.
3. In CNN, some of them are followed by grouping layers and hidden layers are typically convolutional layers followed by activation layers.
4. The pre-processing needed in a ConvNet is kindred to that of the related pattern of neurons in the human brain and was motivated by the organization of the Visual Cortex.

Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data by reducing the dimensions. There are two types of pooling: average pooling and max pooling.

1. Max Pooling: Here we find the maximum value of a pixel from a portion of the image covered by the kernel. It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction.
2. Average Pooling: Here it returns the average of all the values from the portion of the image covered by the Kernel. Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism.



Despite the power and resource complexity of CNNs, they provide in-depth results. At the root of it all, it is just recognizing patterns and details that are so minute and inconspicuous that it goes unnoticed to the human eye. But when it comes to understanding the contents of an image it fails.

**Code and Output**:

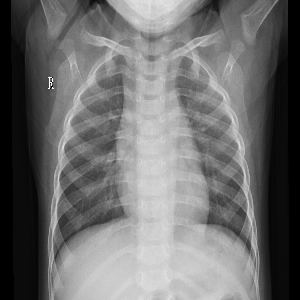
The 2019 novel coronavirus (COVID-19) presents several unique features. While the diagnosis is confirmed using polymerase chain reaction (PCR), infected patients with pneumonia may present on chest X-ray and computed tomography (CT) images with a pattern that is only moderately characteristic for the human eye Ng, 2020.

COVID-19’s rate of transmission depends on our capacity to reliably identify infected patients with a low rate of false negatives. In addition, a low rate of false positives is required to avoid further increasing the burden on the healthcare system by unnecessarily exposing patients to quarantine if that is not required.

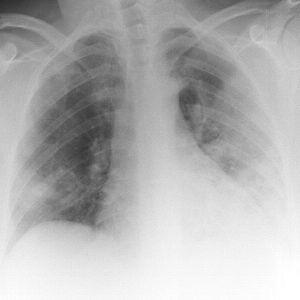
Along with proper infection control, it is evident that timely detection of the disease would enable the implementation of all the supportive care required by patients affected by COVID-19.

| import numpy as np *# Linear algebra* import pandas as pd *# Data processing, CSV file I/O (e.g. pd.read\_csv)* !unzip archive.zip  import tensorflow as tf import matplotlib.pyplot as plt from keras.preprocessing.image import ImageDataGenerator  train\_dir='xray\_dataset\_covid19/train' test\_dir='xray\_dataset\_covid19/test' |
| --- |

| tf.keras.preprocessing.image.load\_img(  'xray\_dataset\_covid19/test/NORMAL/NORMAL2-IM-0035-0001.jpeg', grayscale=True, color\_mode="grayscale", target\_size=(300, 300), interpolation="nearest" ) |
| --- |



| tf.keras.preprocessing.image.load\_img(  'xray\_dataset\_covid19/test/PNEUMONIA/SARS-10.1148rg.242035193-g04mr34g07a-Fig7a-day5.jpeg', grayscale=True, color\_mode="grayscale", target\_size=(300, 300), interpolation="nearest" ) |
| --- |



Data Preprocessing (Augmentation)

| train\_datagen = ImageDataGenerator(rescale=1.0/255, shear\_range=0.1, zoom\_range=0.2, horizontal\_flip=True, featurewise\_center=True, featurewise\_std\_normalization=True, width\_shift\_range=0.10, height\_shift\_range=0.10) training\_set = train\_datagen.flow\_from\_directory(train\_dir, target\_size=(1000,1000), batch\_size=32, color\_mode='grayscale', class\_mode='binary') |
| --- |



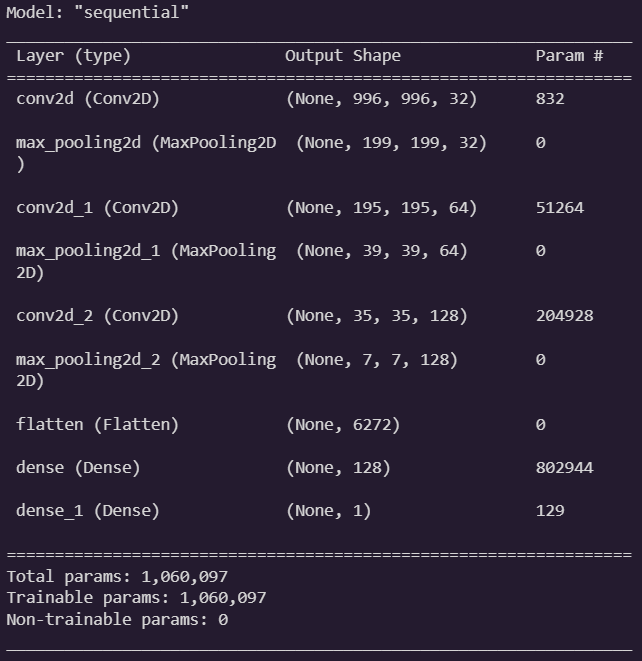
| test\_datagen = ImageDataGenerator(rescale=1.0/255) test\_set = train\_datagen.flow\_from\_directory(test\_dir, target\_size=(1000,1000), batch\_size=32, color\_mode='grayscale', class\_mode='binary') |
| --- |



| from tensorflow.keras.callbacks import EarlyStopping callbacks = EarlyStopping(monitor="val\_acc", patience=3, verbose=1, mode="auto", baseline=None, restore\_best\_weights=False) |
| --- |

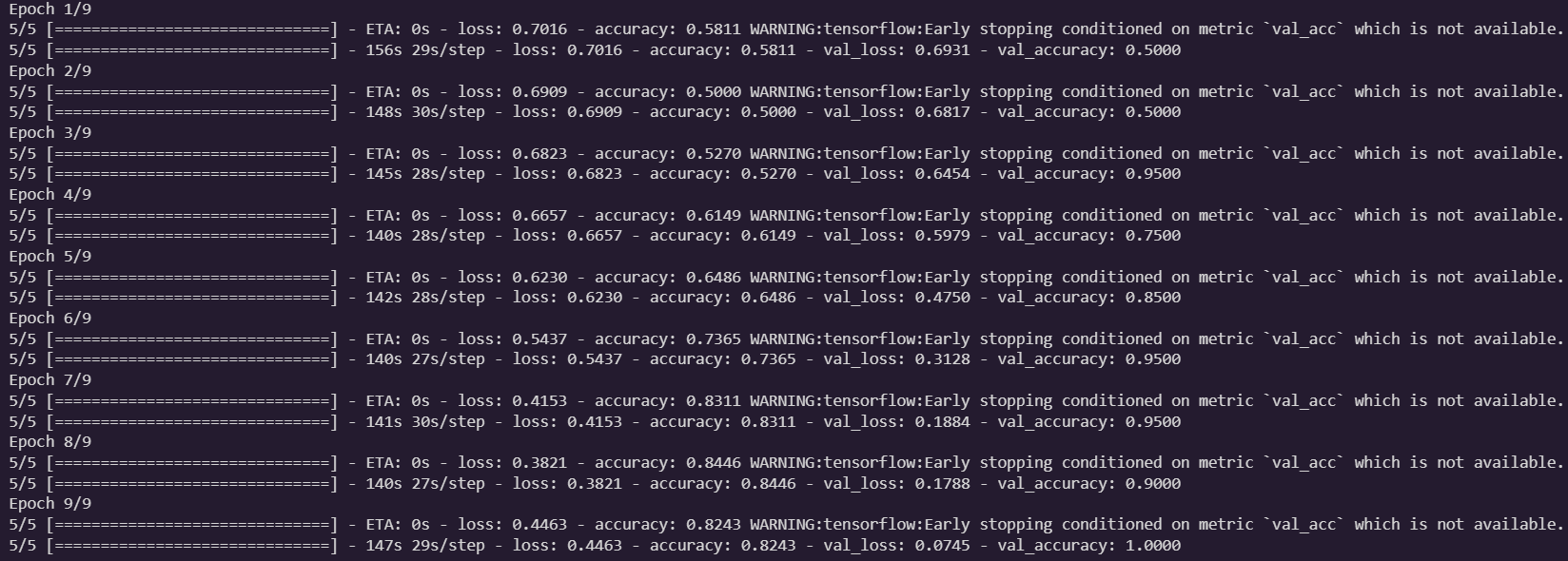
Building the CNN

| *# Initializing CNN* cnn = tf.keras.models.Sequential() *#------------------------------1st Layer------------------------------#* *# Convolution* cnn.add(tf.keras.layers.Conv2D(filters=32, kernel\_size=5, activation='relu', input\_shape=[1000, 1000, 1])) *# Pooling* cnn.add(tf.keras.layers.MaxPool2D(pool\_size=5, strides=5)) *#------------------------------2nd Layer------------------------------#* *# Convolution* cnn.add(tf.keras.layers.Conv2D(filters=64, kernel\_size=5, activation='relu')) *# Pooling* cnn.add(tf.keras.layers.MaxPool2D(pool\_size=5, strides=5)) *#------------------------------3rd Layer------------------------------#* *# Convolution* cnn.add(tf.keras.layers.Conv2D(filters=128, kernel\_size=5, activation='relu')) *# Pooling* cnn.add(tf.keras.layers.MaxPool2D(pool\_size=5, strides=5)) *# Flattening* cnn.add(tf.keras.layers.Flatten()) *# Full Connection* *# Hidden Layers* cnn.add(tf.keras.layers.Dense(units=128, activation='relu')) *# Output Layer* cnn.add(tf.keras.layers.Dense(units=1, activation='sigmoid')) cnn.summary() |
| --- |

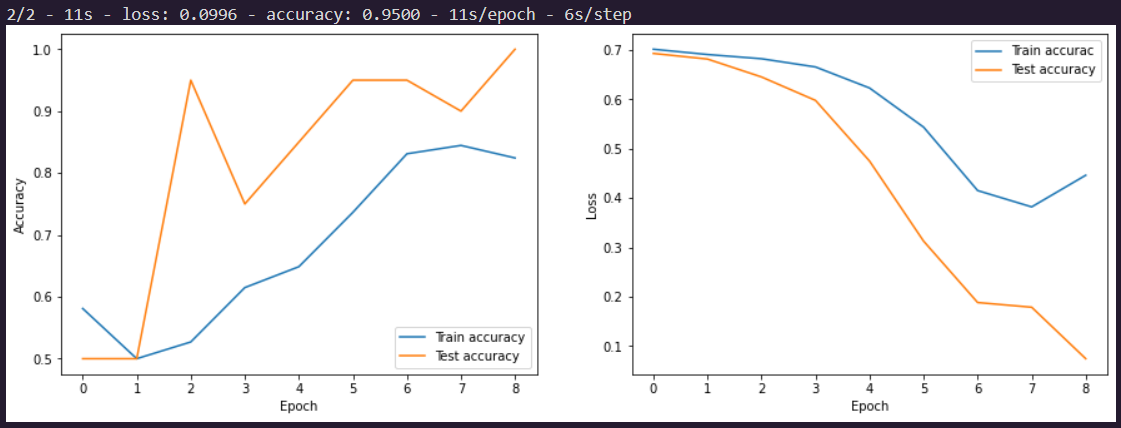


Compiling, Training and Evaluating the model

| *# Compiling* cnn.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy']) *# Training and evaluating* output = cnn.fit(x=training\_set, validation\_data=test\_set, epochs=9, callbacks=callbacks) |
| --- |



| plt.figure(figsize=(15, 5))  plt.subplot(1, 2, 1) plt.plot(output.history['accuracy'], label='accuracy') plt.plot(output.history['val\_accuracy'], label='val\_accuracy') plt.xlabel('Epoch') plt.ylabel('Accuracy') plt.legend(['Train accuracy', 'Test accuracy'], loc='lower right')  plt.subplot(1, 2, 2) plt.plot(output.history['loss'], label='loss') plt.plot(output.history['val\_loss'], label='val\_loss') plt.xlabel('Epoch') plt.ylabel('Loss') plt.legend(['Train accuracy', 'Test accuracy'], loc='upper right')  test\_loss, test\_acc = cnn.evaluate(test\_set, verbose=2) |
| --- |



| image = tf.keras.preprocessing.image.load\_img('xray\_dataset\_covid19/test/PNEUMONIA/ryct.2020200034.fig2.jpeg',target\_size=(1000,1000), grayscale=True, color\_mode="grayscale") input\_arr = tf.keras.preprocessing.image.img\_to\_array(image) input\_arr = np.array([input\_arr]) *# Convert a single image to a batch.* if cnn.predict(input\_arr) == 1:  print ("Omicron Positive") else:  print ("Omicron Negative") |
| --- |



**Conclusion**:

Thus we studied an overview of the Convolution Neural Network model and how to implement a Deep Learning Application using CNN.